Extending the Exact Minimum Line Cover of Prime-Indexed Points to $N = 1814$

Jesper Gran Mikkelsen
Independent Researcher, Norway
jesgranbusiness@gmail.com

September 2026

Abstract

Place the N prime points $(1, p_1), (2, p_2), \ldots, (N, p_N)$ in the plane, where p_k is the k-th prime, and let $f(N)$ be the minimum number of straight lines covering all N points — OEIS A373813. An earlier paper [1] certified $f(1024) = 143$ using `primecover1024.cpp`, extending the prior certified boundary from $N = 861$ in under 40 hours.

This supplement reports a further extension of that record to $\mathbf{f(1814) = 229}$, adding **790** newly certified, strictly optimal terms ($N = 1025$ through $N = 1814$) and **86** new awkward primes (OEIS A393445), from $N = 1063$ ($p = 8527$, $f = 144$) through $N = 1811$ ($p = 15511$, $f = 229$). The computation used `primecover2048_line_coordinates.cpp`, a lightly modified build of the published line-coordinate solver whose only substantive change is a widening of the coverage bitmask from 1024 to 2048 bits, plus the bookkeeping needed to keep the wider masks from slowing down the first half of the sweep. No part of the algorithm — the incremental sweep, the three execution modes, the cover-inequality and Lagrangian bounds, the Exclusive Dependency Rule, or the parallel frontier — was changed.

The sweep was launched with a target of $N = 2048$ and ran for 461 h 35 min 22 s (19.2 days) of solver wall-clock on the same Google Cloud `c4d-highcpu-8` instance (8 vCPUs, 15 GB RAM) used for the previous record, before the $300 free-trial credit funding the instance was exhausted and the run stopped at $N = 1814$. The horizon $N = 1814$ is therefore a *budget* boundary, not an architectural one: the solver was still running normally, and the remaining 234 indices up to the 2048-bit ceiling are reachable with more compute and no code change.

A single index dominates the cost: $N = 1811$ alone consumed 240 h 10 min (52.0% of the entire sweep) and 6.13 billion branch-and-bound nodes. Across the full sweep the mode split is 1,141 witness hits (`W`), 362 root closures (`R`) and 311 full searches (`D`); the 311 `D` instances account for 99.87% of wall-clock time. The decision space grows from $|A_{1024}| = 12{,}162$ active heavy lines to $|A_{1814}| = 30{,}646$, and the maximum DFS depth rises from 108 to 182. As an independent check, the run reproduces all 1024 previously published terms exactly, with zero mismatches.

MSC 2020 Subject Classification: 05B40, 11A41, 68W40, 90C57.

Keywords: prime points, minimum line cover, set cover, branch-and-bound, awkward primes, OEIS A373813, OEIS A393445.

1 What is new

Given the points $\mathcal{S}_N = \{(i, p_i) : 1 \le i \le N\}$, where p_i is the i-th prime, $f(N)$ denotes the least number of straight lines whose union contains $\mathcal{S}_N$. Three or more of these points are

collinear only when their prime values satisfy an exact arithmetic coincidence, and such coincidences thin out as N grows, which is what makes $f(N)$ hard to certify. The sequence is non-decreasing and almost everywhere flat; the rare indices at which $f(N) > f(N-1)$ are the *awkward primes* [4, 6]. Certifying a record means proving optimality at every index up to the new boundary, not merely at the boundary itself.

The certified boundary has moved twice. It stood at $N = 861$, reached by a general-purpose MIP solver in 282 h 26 min 31.5 s [2, 3]. The paper accompanying this one [1] moved it to $N = 1024$ with a purpose-built incremental solver, certifying $f(1024) = 143$ in under 40 hours. That paper closed by noting that 1024 was an *architectural* ceiling — the width in bits of the coverage bitmask, `kBitCapacity` — and that going further required widening the masks to 2048 bits.

This supplement reports the result of doing exactly that. The headline outcome is $f(1814) = 229$, with all 790 intermediate terms certified optimal. Table 1 summarises the change.

Table 1: The record before and after this run. Timings for the $N \le 861$ and $N \le 1024$ columns are the *same-solver* timings quoted in the respective source; the $N \le 1814$ column is this run.

	Prior MIP	2026 paper	This run
Certified boundary N	861	1024	**1814**
p_N	6679	8161	**15551**
$f(N)$	123	143	**229**
New terms certified	—	163	**790**
New awkward primes	—	20	**86**
Solver	Sage/MIP	`primecover1024.cpp`	`primecover2048_...cpp`
Mask width	—	1024 bit	2048 bit
Wall-clock	282 h	<40 h	461.6 h
Stopped by	completion	completion	**credit exhaustion**

How the run ended

The sweep was configured for $N_{\text{end}} = 2048$ and launched on a Google Cloud `c4d-highcpu-8` instance (8 vCPUs, 15 GB RAM, Zen 5), the same instance shape used for the $N = 1024$ record, funded by Google Cloud's $300 free-trial credit. It ran continuously for 461 h 35 min 22 s — just over 19 days — and certified $N = 1814$ at $p_{1814} = 15551$ before the credit ran out and the instance was stopped.

The run was healthy at the moment it ended. $N = 1814$ closed normally in 2,312.7 s, and the final three indices ($N = 1812, 1813, 1814$) took 0.06 s, 5,176.8 s and 2,312.7 s respectively — fast by the standards of that part of the sweep. Nothing about 1814 is special; it is simply where the money stopped. The 2048-bit architectural ceiling is still 234 indices away, and reaching it requires no source change — only more compute, and substantially more of it, since the per-index cost in this regime is growing far faster than linearly (figure 6).

2 The solver: `primecover2048_line_coordinates.cpp`

The solver for this run is a lightly modified copy of the line-coordinate variant described in [1], `primecover1024_line_coordinates.cpp`. It is placed at the repository root alongside the original. A full textual diff between the two files touches 114 lines — 78 added, 36 removed — of the 3,432 in the original, and table 2 lists every substantive change.

Table 2: Every substantive difference between the two files. Items 2–4 exist only to stop the wider mask from slowing down the part of the sweep that does not need it; item 5 is the one change that costs measurable performance.

Change	What and why
1. Mask width	`kBitCapacity` 1024 → 2048, so `kBitWords` 16 → 32 and the type `BitMask1024` becomes `BitMask2048`. Each heavy line's coverage mask grows from 128 to 256 bytes.
2. Live-prefix `and_not`	A `used_words` overload of `BitMask::and_not` loops over only the live word prefix instead of all 32 words, together with an uninitialised-construction tag that skips zeroing the untouched upper words.
3. Live-prefix state keys	`StateKey` carries `used_words`, and both equality and hashing are restricted to that prefix. Without this, every frontier dominance-map lookup would hash 32 words instead of 16 for the whole first half of the sweep.
4. Reused scratch mask	`build_compact_productive_lines` writes into a preallocated `productive_union_scratch` cleared over `words_` words only, replacing a 256-byte zero-init that fired on every DFS node. (Adds `#include <cstring>`.)
5. Frontier ladder cap	The top rung of the frontier multiplier $m(C)$ is commented out: the $C \geq 128 \Rightarrow 8192U$ rung is disabled, leaving $C \geq 125 \Rightarrow 2048U$ as the maximum. On 8 workers this caps the parallel frontier at $8 \times 2048 = 16{,}384$ tasks rather than 65,536. With masks twice as wide, per-task memory roughly doubles, and the old top rung no longer fits in 15 GB.

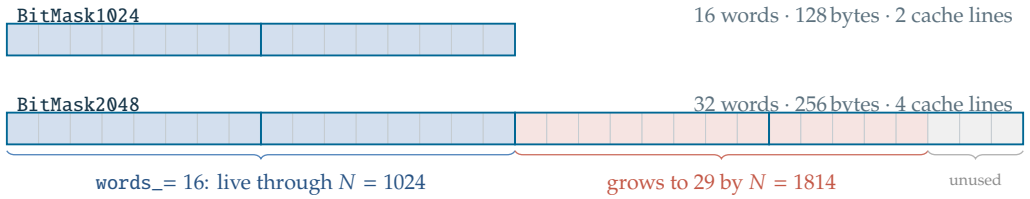

Figure 1: The one structural change. Each heavy line's coverage mask grows from sixteen `uint64_t` words to thirty-two; bit k still records whether zero-based prime index k lies on the line. Because the sweep is incremental, only the first `words_` $= \lceil N/64 \rceil$ words are ever live — 16 at $N = 1024$, 29 at $N = 1814$ — so three of the thirty-two words are never touched in this run. Items 2–4 of table 2 exist to make `and_not`, state-key hashing and the per-node scratch clear respect that prefix; without them the entire first half of the sweep would pay for the upper sixteen words it never reads, which is why the wider build is only 1% slower over $N \leq 861$ (table 3).

Nothing in the list touches the algorithm. The incremental sweep, the witness mechanism, the three execution modes, the cover-inequality floor $\lceil |S|/2 \rceil$, the Lagrangian gain bound with its projected subgradient and coordinate-descent polish, the Exclusive Dependency Rule, strong branching, and the parallel frontier with dominance pruning are all byte-for-byte identical to the published solver. Consequently every proof in [1] — in particular the exchange argument justifying the Exclusive Dependency Rule and the mode-by-mode certification guarantee — applies to this run unchanged.

What the modifications cost

Because the run re-sweeps $N = 1, \ldots, 1024$ from scratch, the two builds can be compared directly on identical instances. Table 3 does this; its `1024` column is the published world-record run of `primecover1024.cpp`, which differs from the line-coordinate variant this build was forked from only in what it prints.

Table 3: The 2048-bit build measured against the published 1024-bit build on the same indices and the same instance shape. The last column is the largest frontier reached at a mode-`D` row of each build. Below $C = 128$ the two ladders are identical, so nothing in $N \le 861$ changes; of the old build's fourteen `D` rows in $N = 862$–892 only two reached the disabled top rung, whereas in $N = 893$–1024 all thirty-nine did. The regression sits exactly there.

Index range	`1024` build	`2048` build	Ratio	max frontier, `1024` → `2048`
$N = 1$–861	1,355.9 s	1,366.7 s	1.01×	4,096 → 4,096
$N = 862$–892	12,762.9 s	10,120.5 s	0.79×	65,536 → 16,384
$N = 893$–1024	128,846.2 s	159,577.9 s	1.24×	65,536 → 16,384
$N = 1$–1024	142,964.9 s (39.71 h)	171,065.1 s (47.52 h)	1.20×	

Doubling the mask width is close to free: over $N \le 861$, where the frontier ladder is identical in both builds, the wider build is 1% slower. That is the payoff from items 2–4 of table 2 and of figure 1. The 1.24× regression over $N = 893$–1024 coincides exactly with the range in which every mode-`D` row of the old build used a 65,536-task frontier and the new one uses 16,384 — a fourfold loss of parallel granularity at precisely the hardest instances. In the narrow band $N = 862$–892 only two of the old build's fourteen `D` rows reached the disabled rung — too few for the cap to show through the ordinary run-to-run spread — and the block happens to land 0.79×; that is not evidence that capping helps. The regression is a memory-budget consequence of the 15 GB instance, not of the mask widening, and it would be recovered on a machine with ≥ 30 GB of RAM by re-enabling the 8192U rung.

Build and invocation

The run used GCC 14 targeting Zen 5:

```
g++-14 -std=c++23 -O3 -march=znver5 -pthread -fno-exceptions -fno-rtti
        primecover2048_line_coordinates.cpp -o solver
```

with the solver launched under `nohup` and left to run. Worker count is `std::thread::hardware_concurrency()`, i.e. 8 on this instance.

3 Results

3.1 The staircase and the 86 new awkward primes

Figure 2 plots $f(N)$ over the whole certified range. The curve is a staircase: flat almost everywhere, with a riser exactly at each awkward prime — an index whose new point falls on none of the lines of any optimal cover of its predecessor, forcing the cover to grow by one. Since f counts nothing but those risers, $f(1814) = 229$ is the same statement as "there are 229 awkward primes at or below $N = 1814$". The three colour bands record only which computation first certified each stretch; this run re-derives all of them from scratch.

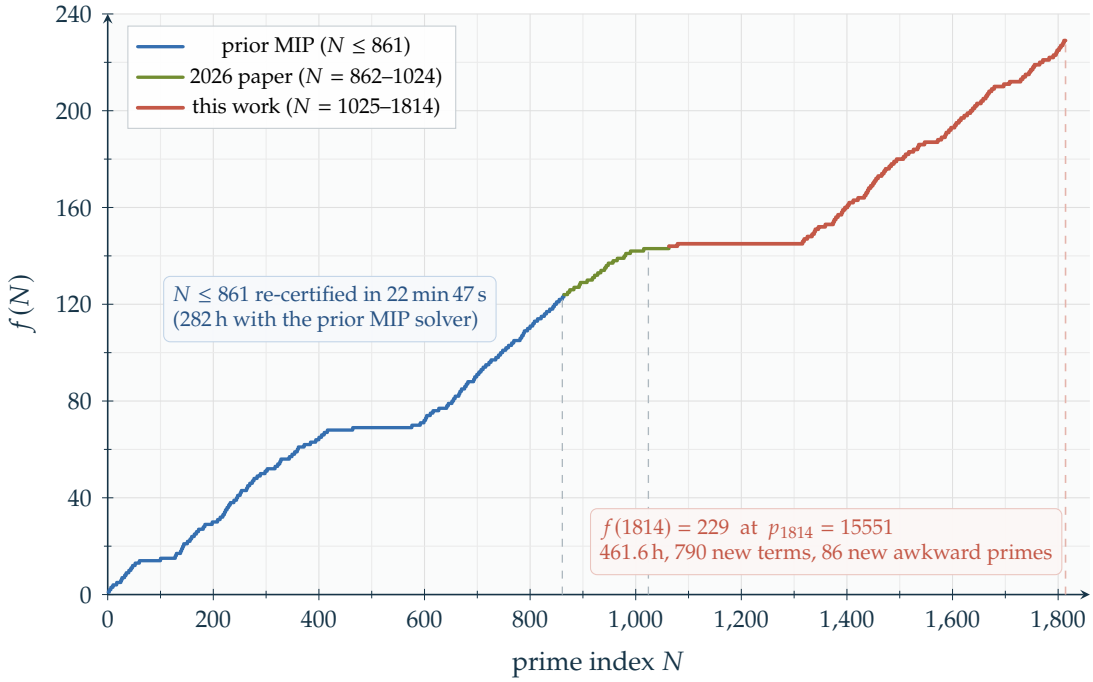

Figure 2: Minimum line cover number $f(N)$ over all 1814 certified indices. Each riser is an awkward prime. The blue segment is the range certified by the prior MIP solver [2] in 282 h 26 min 31.5 s; this run re-certifies it in 1,366.7 s (22 min 47 s), a 744.0× speedup. The olive segment is the 163-term extension of [1]. The red segment is the 790 new terms reported here, raising the record from $f(1024) = 143$ to $f(1814) = 229$ and adding 86 awkward primes (table 4). Sequence: OEIS A373813.

Every riser in the red band is listed in table 4. These 86 indices extend OEIS A393445 from $N = 1015$ to $N = 1811$. They are also thinning: 86 risers among the 790 new indices is one in nine, against 143 among the first 1024, one in seven. That is the expected behaviour — three collinear prime points require an exact arithmetic coincidence, and coincidences become scarcer as the points spread out — but it is worth stating as a measurement rather than an expectation, because it is the first time the range $N > 1024$ has been certified at all.

3.2 How each index was certified

Each index resolves in one of three modes: witness bypass (`W`), root closure (`R`), or full branch-and-bound (`D`). Figure 3 colours the staircase by mode. The proportions hardly move with N: mode `D` fires at 17.1% of the indices over the whole sweep and at 17.1% over the newly certified block taken alone. What changes is not how often the exact solver runs but what it costs when it does, which is the subject of section 3.3.

That the cheap modes keep working at this scale is not obvious, because the space they are avoiding has grown a great deal. A heavy line — one through at least three points of $\mathcal{S}_{N_{\text{end}}}$ — becomes *active* at the step where its third-smallest point index is reached, and the count $|A_N|$ of active heavy lines is the exact size of the solver's decision space at step N (figure 4). It ends the sweep 2.52× above the ceiling of the previous record, and most steps still never look at it: the witness cover carried forward from step $N - 1$ already covers the new point, and the step closes without the exact solver being entered.

3.3 Where the 19 days went

The three modes differ in cost by orders of magnitude, and the gap widens with N. Figure 5 plots every index individually.

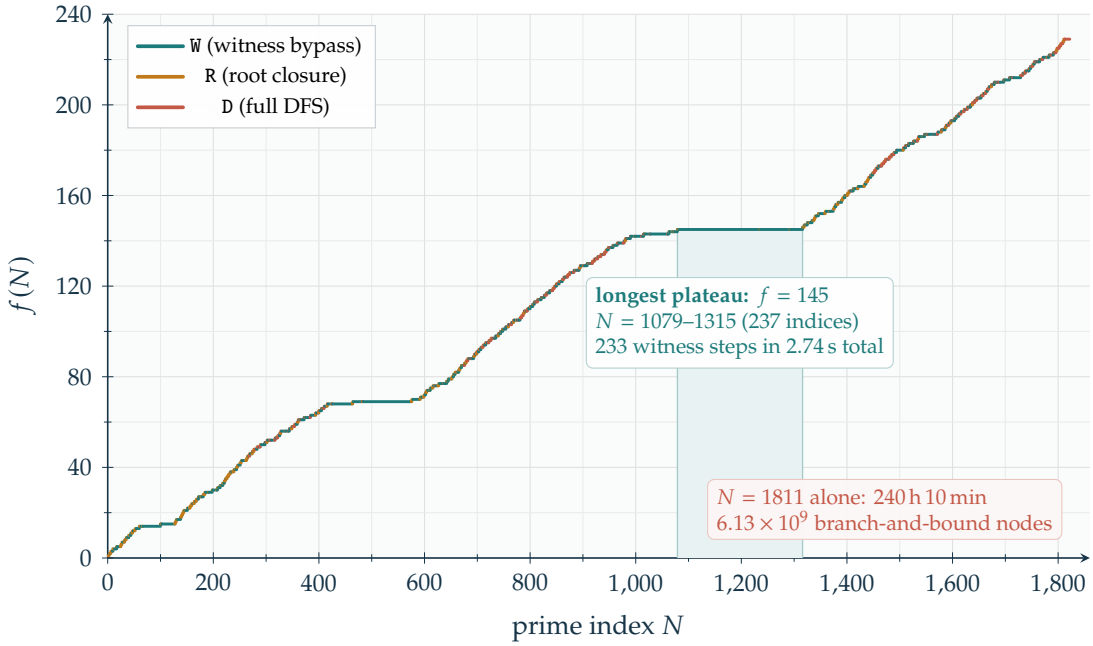

Figure 3: The staircase coloured by how optimality was certified at each step. Across all 1814 indices the split is 1,141 witness hits (62.9%, 10.2 s total), 362 root closures (20.0%, 2,176.1 s total) and 311 full searches (17.1%, 1,659,535.7 s total) — so 17.1% of the indices consume 99.87% of the wall-clock. The shaded band is the new longest plateau: $f = 145$ holds for 237 consecutive indices, $N = 1079$ through $N = 1315$, of which 233 are certified by witness bypass in 2.74 s in total; $p_{1316} = 10831$ then falls on no existing cover line and forces $f \to 146$. This plateau is more than twice the length of the 112-index run at $f = 69$ ($N = 464$–575) featured in [6], whose closer $p_{576} = 4211$ keeps its name as the *party-pooper prime*; $p_{1316} = 10831$ is now the closer of the longest known plateau.

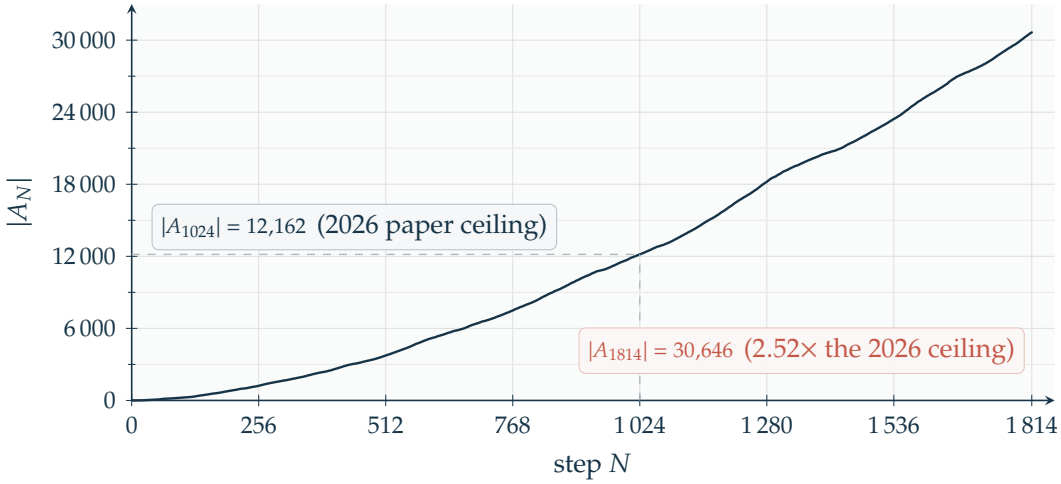

Figure 4: Active heavy lines $|A_N|$ at each step $N = 1, \ldots, 1814$. The count grows super-linearly, from 9,240 at $N = 861$ and 12,162 at $N = 1024$ to 22,565 at $N = 1500$ and 30,646 at $N = 1814$ — a 2.52× increase over the range added by this run. At 256 bytes per mask the working set of heavy-line masks is now 7.85 MB, five times the 1.56 MB the published 1024-bit build held at $N = 1024$; it is well past any per-core L2 and a visible fraction of shared L3, which is part of why per-index cost grows faster than $|A_N|$ alone would suggest. The witness mechanism absorbs the growth almost entirely: even with more than 30,000 active lines in the final stretch, most steps still resolve in mode `W` without invoking the exact solver at all.

Table 4: The 86 new awkward primes, $N = 1063$ through $N = 1811$: indices at which $f(N) > f(N-1)$, with the prime p_N and the resulting cover size $f(N)$. Read down each column group, then across. Together with the 143 awkward primes at $N \le 1024$ these give 229 in total — which is also, by construction, the value of $f(1814)$.

N	p_N	$f(N)$	N	p_N	$f(N)$	N	p_N	$f(N)$
1063	8527	144	1459	12203	173	1642	13901	202
1079	8669	145	1466	12263	174	1646	13921	203
1316	10831	146	1470	12289	175	1654	14011	204
1319	10853	147	1473	12329	176	1658	14057	205
1325	10891	148	1480	12401	177	1663	14107	206
1334	10987	149	1484	12433	178	1666	14153	207
1338	11047	150	1489	12479	179	1669	14177	208
1340	11059	151	1494	12511	180	1674	14249	209
1346	11113	152	1507	12613	181	1679	14321	210
1359	11243	153	1511	12647	182	1697	14479	211
1374	11383	154	1517	12703	183	1708	14563	212
1376	11399	155	1526	12799	184	1729	14759	213
1379	11437	156	1534	12893	185	1733	14783	214
1383	11471	157	1536	12907	186	1738	14831	215
1390	11527	158	1547	12983	187	1745	14891	216
1392	11551	159	1572	13219	188	1748	14929	217
1396	11597	160	1579	13297	189	1752	14957	218
1401	11677	161	1586	13367	190	1755	15013	219
1404	11699	162	1588	13397	191	1765	15107	220
1412	11783	163	1593	13441	192	1771	15161	221
1421	11839	164	1597	13469	193	1783	15271	222
1433	11953	165	1604	13553	194	1791	15329	223
1436	11971	166	1607	13591	195	1796	15373	224
1439	12007	167	1612	13633	196	1798	15383	225
1441	12037	168	1616	13681	197	1802	15427	226
1445	12071	169	1622	13711	198	1805	15451	227
1449	12107	170	1628	13759	199	1809	15493	228
1452	12119	171	1634	13829	200	1811	15511	229
1455	12157	172	1638	13873	201			

Because a handful of indices dominate, no average over the sweep says anything useful about it. Table 5 lists the five costliest instances, and figure 6 accumulates the whole log.

Table 5: The five costliest instances of the sweep. Together they consume 71.5% of the 461.6 hours; $N = 1811$ alone consumes 52.0%. All five resolve in mode `D`, and all five lie in the block added by this run.

N	p_N	$f(N)$	wall-clock (h)	share	B&B nodes	depth
1811	15511	229	240.17	52.03%	6,125,319,324	175
1809	15493	228	29.45	6.38%	700,457,753	148
1679	14321	210	29.06	6.30%	1,209,486,932	149
1674	14249	209	15.87	3.44%	1,170,054,358	147
1755	15013	219	15.28	3.31%	459,489,183	156

3.4 Aggregate search statistics

Table 6 sums the per-index counters, once over the whole sweep and once over the newly certified block alone. The second column is the honest measure of what this run added: the

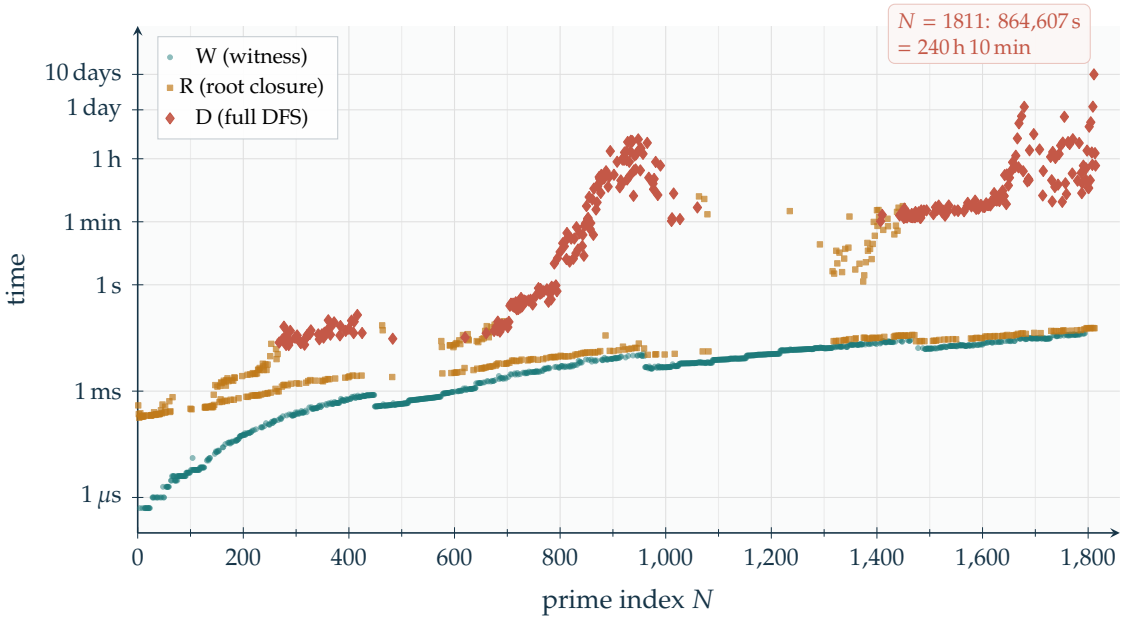

Figure 5: Per-instance solve time on a logarithmic scale across all 1814 certified indices, by execution mode: circles are witness hits (`W`), squares are root closures (`R`), diamonds are full searches (`D`). The spread now covers close to twelve orders of magnitude, from microsecond witness hits to the ten-day solve at $N = 1811$. Witness and root rows stay essentially flat across the whole sweep — their cost tracks $|A_N|$, not the difficulty of the instance — while the `D` envelope climbs steeply past $N \approx 1600$. Rows logged as 0.000000 s are floored at 0.5 μs for plotting.

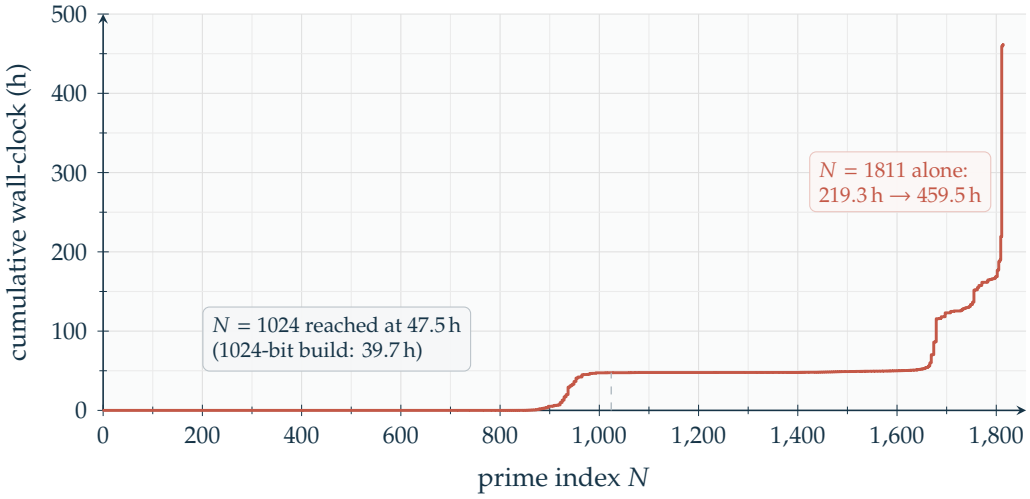

Figure 6: Cumulative solver wall-clock against N. The first 1024 indices — the entire previous record — account for 47.5 of the 461.6 hours, about 10%. The curve is almost flat until $N \approx 1600$ and then rises sharply; the single vertical jump at $N = 1811$ is one instance consuming 240 h 10 min, 52.0% of the whole run. The $300 credit expired shortly after $N = 1814$.

first 1024 indices were already known, and re-deriving them cost about a tenth of the budget.

Table 6: Aggregate counters over the full sweep and over the newly certified block, summed from the per-index log rows.

	$N = 1$–1814	$N = 1025$–1814
Indices certified	1,814	790
Wall-clock	1,661,722.0 s	1,490,656.8 s
	(461.59 h)	(414.07 h)
Mode `W` / `R` / `D`	1,141 / 362 / 311	528 / 127 / 135
Branch-and-bound nodes	22,669,986,892	14,364,202,147
Lagrangian prunes	6,267,952,704	2,395,201,989
Exclusive-Dependency forcings	53,407,641	23,023,450
Strong-branching invocations	4,087	2,576
Maximum DFS depth	182 (at $N = 1814$)	182

4 Quantities in the 2026 paper that now need updating

The architecture description in [1] remains accurate; what has changed are the numbers instantiating it. Table 7 lists every quantity in that paper whose value this run supersedes, so each can be carried across without re-deriving it.

Three entries deserve a note.

The frontier ladder. The piecewise-constant multiplier $m(C)$ published in [1] had six rungs with $m(C) = 8192$ for $C \geq 128$. For this run the top rung is disabled, so the ladder effectively reads

$$m(C) = \begin{cases} 2048 & C \geq 125, \\ 512 & 121 \leq C < 125, \\ 128 & 113 \leq C < 121, \\ 16 & 93 \leq C < 113, \\ 4 & C < 93, \end{cases}$$

and the largest frontier observed in the log is 16,384 tasks, against 65,536 in the published run. This is a memory constraint of the 15 GB instance, not a correctness one: any sufficiently large frontier yields a provably optimal answer, and a smaller one merely serialises work.

Cache residency. The claim in [1] that the full mask working set stays L2-resident throughout the sweep no longer holds at this scale. At 30,646 active lines and 256 bytes per mask the set is 7.85 MB, comfortably past the per-core L2 of the target CPU, though still small against L3. The popcount-over-word-wise-AND formulation is unaffected; only the residency claim needs softening.

Mode assignment is build-dependent. Over the 1024 indices covered by both runs, the certified values $f(N)$ agree at every single index, but the recorded *mode* differs at 7 of them. This is expected: which of `W`, `R` and `D` fires at a given step depends on the warm state and frontier configuration inherited from earlier steps, neither of which is part of the optimality certificate. The same caveat applies to search-shape counters: over $N \leq 1024$ this run reaches DFS depth 112 where the published run reached 108, on identical and identically certified

Table 7: Quantities from [1] superseded by this run. Values in the middle column are as published; values in the right column are measured from the $N = 1814$ sweep log.

Quantity	**As published ($N = 1024$)**	**Now ($N = 1814$)**
Record and sequence		
Certified boundary	$N = 1024$, $f = 143$	$N = 1814$, $f = 229$
Prime at the boundary	$p_{1024} = 8161$	$p_{1814} = 15551$
New A373813 terms	163 (862–1024)	790 (1025–1814)
New A393445 terms	20 (864–1015)	86 (1063–1811)
Total awkward primes $\leq N$	143	229
Data structure		
`kBitCapacity`	1024	2048
`kBitWords`	16	32
Bytes per heavy-line mask	128 (2 cache lines)	256 (4 cache lines)
Mask working set	$\approx$ 1.56 MB (L2-resident)	$\approx$ 7.85 MB (exceeds per-core L2)
Search space		
Active heavy lines at boundary	$\lvert A_{1024}\rvert = 12{,}162$	$\lvert A_{1814}\rvert = 30{,}646$
$\lvert A_N\rvert$ at awkward primes	$\approx$9,000–12,000	up to 30,646
Maximum DFS depth	108	182
Node counts at hardest instances	hundreds of millions	up to 6.13×10^9
Execution profile		
Mode split (count)	615 / 236 / 173	1141 / 362 / 311
Mode split (share)	60 / 23 / 17 %	62.9 / 20.0 / 17.1 %
`D` share of wall-clock	99.998%	99.87%
Total `D` time	142,961.9 s	1,659,535.7 s
Grand total	142,965 s (39.7 h)	1,661,722 s (461.6 h)
Time to $N = 861$	1,355.9 s (749.9× vs MIP)	1,366.7 s (744.0× vs MIP)
Parallel frontier		
Top rung of $m(C)$	8192 for $C \geq 128$	2048 for $C \geq 125$ (top rung disabled)
Maximum frontier tasks	$8 \times 8192 = 65{,}536$	$8 \times 2048 = 16{,}384$
Plateaus		
Longest plateau	$f = 69$, $N = 464$–575 (112)	$f = 145$, $N = 1079$–1315 (237)
Plateau closer	$p_{576} = 4211$	$p_{1316} = 10831$
Filename semantics		
Meaning of the numeral	1024 = architectural ceiling, also where the sweep ended	2048 = architectural ceiling; the sweep ended at 1814 for budget reasons

instances. Mode counts and node, depth or prune totals quoted from either run should be read as properties of that run, not of the sequence.

5 Verification and data

Cross-check against the published record. The sweep recomputes $N = 1, \ldots, 1024$ from scratch under a different mask width, a different frontier ladder and a different hot-path memory layout. Comparing its output against the published `B373813.txt` gives 1024 **of** 1024 **indices matching, with zero mismatches**. Since the two builds share no state and differ in exactly the ways listed in table 2, this is a genuine independent revalidation of the earlier record rather than a re-run of the same computation.

Optimality. Every reported $f(N)$ is proven optimal. The solver terminates only when the gap between its combined lower bound and its incumbent reaches zero; there is no heuristic stopping rule, and no index in this sweep was abandoned on a timeout.

Files. The sweep log, the derived sequence files, and the solver are in the repository [7]:

- `primecover2048_line_coordinates.cpp` — the solver that produced this record, at the repository root;
- the full sweep log for $N = 1$–1814, including the line coordinates of every optimal cover, which is the authoritative record for all 790 new terms;
- the two-column sequence files for A373813 (N, $f(N)$) and A393445 (the awkward primes of table 4).

Every figure, and every number describing *this* run, is derived mechanically from that one sweep log; the published log alone regenerates them. The comparison columns — the `1024` build in table 3 and the *as published* column of table 7 — come from the previous run's own log and from [1]. The 1024-bit release this build was forked from is archived on Zenodo [5].

`https://github.com/jespergran98/prime-line-cover`

What it would take to finish. Reaching the 2048-bit ceiling means certifying 234 further indices in a regime where single instances already cost ten days. Extrapolating figure 6 is unwise — the cost is driven by individual hard instances rather than by a smooth trend — but the run makes one thing concrete: a machine with ≥ 30 GB of RAM would restore the 65,536-task frontier, and on the evidence of table 3 that alone is worth roughly 20% of wall-clock in the hard regime.

References

[1] J. G. Mikkelsen, *An Exact Solver for the Minimum Line Cover of Prime-Indexed Points*, 2026. Included in the repository below; certifies $f(1024) = 143$.

[2] M. A. Alekseyev, *Sage program for lines covering points*, GitHub, August 2024. `https://github.com/maxale/oeis/`. See also OEIS A373813 contributor notes, `https://oeis.org/A373813`.

[3] OEIS Foundation Inc., *The On-Line Encyclopedia of Integer Sequences*, A373813, 2026. `https://oeis.org/A373813`

[4] OEIS Foundation Inc., *The On-Line Encyclopedia of Integer Sequences*, A393445, 2026. https://oeis.org/A393445

[5] J. G. Mikkelsen, *prime-line-cover: Exact Minimum Line Cover Solver for Prime Points*, Zenodo, 2026. https://doi.org/10.5281/zenodo.20096556

[6] B. Haran (producer), *Awkward Primes* (featuring N. Sloane), Numberphile, April 7, 2026. https://www.youtube.com/watch?v=VFoIPlUalRY

[7] J. G. Mikkelsen, *prime-line-cover: Exact Minimum Line Cover Solver for Prime Points (source code, results, and interactive demo)*, GitHub, 2026. https://github.com/jespergran98/prime-line-cover. Interactive demo: https://prime-line-cover.vercel.app.